**TRƯỜNG ĐẠI HỌC TÀI NGUYÊN VÀ MÔI TRƯỜNG TP.HCM**

**KHOA: HỆ THỐNG THÔNG TIN VÀ VIỄN THÁM**



**BÁO CÁO MÔN HỌC : CÔNG NGHỆ LẬP TRÌNH ĐA NỀN TẢNG CHO ỨNG DỤNG DI ĐỘNG**

**ĐỀ TÀI : FLUTTER TO - DO APP**

Giảng viên hướng dẫn : **Nguyễn Thanh Truyền**

# Sinh viên thực hiện : **Phạm Từ Khôi Nguyên - 0850080086**

# **Nguyễn Ngọc Gia Huy - 0850080022**

# **Huỳnh Huy Anh Hào - 0850080070**

# **Phan Nguyễn Thành Nhân - 0850080089**

# **Võ Khánh Tâm - 0850080103**

# **Lê Chí Thành - 0850080104**

# Lớp : **08\_ĐH\_THMT**

# Khoá : **08\_HTTT & VT**

***TP. Hồ Chí Minh, tháng 8 năm 2023***

**TRƯỜNG ĐẠI HỌC TÀI NGUYÊN VÀ MÔI TRƯỜNG TP.HCM**

**KHOA: HỆ THỐNG THÔNG TIN VÀ VIỄN THÁM**



**BÁO CÁO MÔN HỌC : CÔNG NGHỆ LẬP TRÌNH ĐA NỀN TẢNG CHO ỨNG DỤNG DI ĐỘNG**

**ĐỀ TÀI : FLUTTER TO - DO APP**

Giảng viên hướng dẫn : **Nguyễn Thanh Truyền**

# Sinh viên thực hiện : **Phạm Từ Khôi Nguyên - 0850080086**

# **Nguyễn Ngọc Gia Huy - 0850080022**

# **Huỳnh Huy Anh Hào - 0850080070**

# **Phan Nguyễn Thành Nhân - 0850080089**

# **Võ Khánh Tâm - 0850080103**

# **Lê Chí Thành - 0850080104**

# Lớp : **08\_ĐH\_THMT**

# Khoá : **08\_HTTT & VT**

***TP. Hồ Chí Minh, tháng 8 năm 2023***

**MỞ ĐẦU**

Trong bối cảnh cuộc cách mạng công nghệ 4.0, sự phát triển mạnh mẽ của điện thoại di động và ứng dụng di động đã góp phần thay đổi cách chúng ta tương tác và quản lý cuộc sống hàng ngày. Hiện nay, việc sử dụng ứng dụng di động trở nên phổ biến và không thể thiếu trong việc quản lý công việc, nhiệm vụ, và mục tiêu cá nhân.

Tuy nhiên, việc phát triển ứng dụng di động đa nền tảng đôi khi gặp phải nhiều thách thức. Mỗi hệ điều hành (OS) di động như Android và iOS đều có cấu trúc và ngôn ngữ lập trình riêng biệt, dẫn đến việc phải viết mã nguồn riêng biệt cho từng nền tảng, tiêu tốn thời gian và nguồn lực. Điều này tạo ra một rào cản đối với việc phát triển ứng dụng nhanh chóng và hiệu quả trên nhiều nền tảng.

Nhằm giải quyết vấn đề này và đáp ứng nhu cầu ngày càng cao của người dùng về một ứng dụng di động linh hoạt và dễ sử dụng, nền tảng Flutter đã ra đời. Flutter là một framework mã nguồn mở của Google, cho phép các nhà phát triển xây dựng ứng dụng di động đa nền tảng với một lần lập trình duy nhất. Điều này giúp giảm thiểu thời gian phát triển, tối ưu hóa hiệu suất ứng dụng, và mang đến trải nghiệm tương tự trên các nền tảng khác nhau.

Trong môn học " **Công nghệ lập trình đa nền tảng cho ứng dụng di động** " chúng ta quan tâm đến việc nghiên cứu và phát triển một ứng dụng di động đơn giản, nhưng mạnh mẽ, sử dụng công nghệ Flutter. Đề tài " **Flutter To-Do App** " sẽ tập trung vào xây dựng một ứng dụng quản lý công việc và nhiệm vụ cá nhân, giúp người dùng tối ưu hóa thời gian và nâng cao hiệu suất công việc.

Qua việc nghiên cứu và phát triển ứng dụng " **Flutter To-Do** " chúng ta sẽ tiếp cận và khám phá tiềm năng của Flutter trong việc tạo ra các ứng dụng di động đa nền tảng một cách hiệu quả và tiện lợi, đồng thời đóng góp vào sự phát triển và ứng dụng thực tiễn của công nghệ lập trình đa nền tảng trong thế giới di động ngày nay. **LỜI CẢM ƠN**

Trong quá trình thực hiện báo cáo môn học cuối kỳ, nhóm em đã nhận được sự giúp đỡ rất nhiệt tình của Trường Đại Học Tài Nguyên Và Môi Trường T.PHCM, thầy cô và giáo viên hướng dẫn.

Để có thể hoàn thành tốt báo cáo, nhóm em xin gửi lời cảm ơn chân thành đến ban giám hiệu Trường Đại Học Tài Nguyên Và Môi Trường TPHCM và thầy Nguyễn Thanh Truyền đã tận tình giảng dạy và cung cấp những kiến thức cần thiết, bổ ích, hướng dẫn truyền đạt nhiều kinh nghiệm trong quá trình giảng dạy cũng như trong thời gian giúp nhóm em có thể hoàn thành tốt quá trình học tập và báo cáo của mình.

Nhóm em xin chân thành cảm ơn !!

**NHẬN XÉT**

**( Của giảng viên hướng dẫn tại khoa )**

…………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………

**Kết luận: Đồng ý hoặc Không đồng ý cho sinh viên nộp báo cáo môn học.**

**……………………………………………………………………………………………………**

………….., ngày….tháng….năm……

CÁN BỘ HƯỚNG DẪN

*(ký tên)*

**LỊCH LÀM VIỆC**

Nhóm 2

Họ và tên người hướng dẫn : **Nguyễn Thanh Truyền**

Thời gian thực tập từ ngày 10 tháng 7 năm 2023 đến ngày 10 tháng 8 năm 2023

|  |  |  |  |
| --- | --- | --- | --- |
| Tuần | Nội dung  công việc được giao | Nhận xét của  CB hướng dẫn | Chữ ký của  CB hướng dẫn |
| **1**  Từ ngày 10/7  đến ngày 17/7 | Tìm hiểu về Navigator và Router |  |  |
| **2**  Từ ngày 17/7  đến ngày 24/7 | Nghiên cứu tổng quan :  - Di chuyển dữ liệu giữa các màn hình. - Lập trình bất đồng bộ với Future, async |  |  |
| **3**  Từ ngày 31/7  đến ngày 7/8 | - Vẽ sơ đồ hệ thống. - Thiết kế cơ sở dữ liệu. - Code giao diện đề tài, xử lý code dự án. |  |  |
| **4**  Từ ngày 7/8  đến ngày 10/8 | - Fix Bug - Chạy chương trình - Viết báo cáo môn học |  |  |

TP. Hồ Chí Minh, ngày 10 tháng 8 năm 2023

Giảng viên

**Nguyễn Thanh Truyền**

**MỤC LỤC**

[Chương 1: TỔNG QUAN 1](#_Toc21069)

[1.1. Giới thiệu về đề tài 1](#_Toc31060)

[1.1.1. Sơ lược về flutter : 1](#_Toc4249)

[1.1.2. Dart : 1](#_Toc3074)

[1.2. Mục đích đề tài 2](#_Toc11749)

[1.3. Đối tượng và phạm vi nghiên cứu 2](#_Toc26876)

[1.3.1. Đối tượng nghiên cứu: 2](#_Toc560)

[1.3.2. Phạm vi nghiên cứu: 2](#_Toc26843)

[Chương 2: CƠ SỞ LÝ LUẬN 3](#_Toc10487)

[2.1. Lý thuyết 3](#_Toc11130)

[2.1.1. Tổng quan về Navigator ? 3](#_Toc15839)

[2.1.2. Tổng quan về Router ? 4](#_Toc20883)

[2.1.3. Di chuyển dữ liệu giữa các màn hình ? 5](#_Toc11193)

[2.1.4. Lập trình bất đồng bộ với Future, async, await 6](#_Toc15282)

[2.1.5. Firebase 8](#_Toc22008)

[2.2. Kĩ thuật 9](#_Toc30789)

[2.2.1. Ngôn ngữ sử dụng : 9](#_Toc3825)

[2.2.2. Framework : 9](#_Toc8038)

[2.2.3. Công cụ: 10](#_Toc18693)

[2.2.4. Thư viện giao diện: 10](#_Toc29150)

[2.2.5. Hệ quản trị cơ sở dữ liệu: 10](#_Toc8737)

[Chương 3: CÀI ĐẶT THỬ NGHIỆM 11](#_Toc9273)

[3.1. Nghiên cứu đặc tả chức năng, thiết kế cơ sở dữ liệu 11](#_Toc21594)

[3.1.1. Đặc tả chức năng : 11](#_Toc10923)

[3.1.2. Lược đồ use case: 13](#_Toc11976)

[3.1.3. ERD: 14](#_Toc7088)

[3.2. Chi tiết và demo hình ảnh các chức năng : 15](#_Toc32240)

[3.2.1. Giao diện đăng nhập : 15](#_Toc24121)

[3.2.2. Giao diện trang : 16](#_Toc25202)

[Chương 4: KẾT LUẬN 21](#_Toc30711)

[4.1. Kết quả đạt được: 21](#_Toc23689)

[DANH MỤC TÀI LIỆU THAM KHẢO 22](#_Toc23644)

**DANH MỤC CÁC SƠ ĐỒ HÌNH**

Hình 1 : Flutter………………………………………………………………………..1

Hình 2 : Dart…………………………………………………………………………..1

Hình 3 : Router………………………………………………………………………..4

Hình 4 : Future, async, await………………………………………………………….6

Hình 5 : Firebase………………………………………………………………………8

Hình 6 : Lược đồ Use - case………………………………………………………….12

Hình 7 : ERD…………………………………………………………………………13

Hình 8 : Giao diện trang chủ…………………………………………………………15

Hình 9 : Giao diện trang danh sách công việc……………………………………….16

Hình 10 : Giao diện trang Authentication……………………………………………17

Hình 11 : Giao diện trang Firestore Database………………………………………..17

Hình 12 : Giao diện trang thêm công việc…………………………………………...18

Hình 13 : Giao diện trang chi tiết công việc…………………………………………19

Hình 14 : Giao diện sửa thông tin công việc…………………………………………20

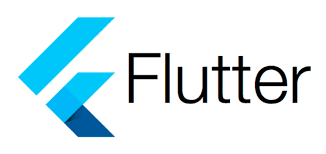
# Chương 1: TỔNG QUAN

## **Giới thiệu về đề tài**

### Sơ lược về flutter :

- Flutter là một framework mã nguồn mở được phát triển bởi Google, cho phép xây dựng ứng dụng di động, máy tính bảng và máy tính từ cùng một mã nguồn duy nhất. Đặc điểm nổi bật của Flutter là khả năng phát triển ứng dụng nhanh chóng và đẹp mắt, nhờ vào tích hợp một tập lệnh gọi là Dart, đồng thời cung cấp các thành phần giao diện người dùng (UI) tùy chỉnh.

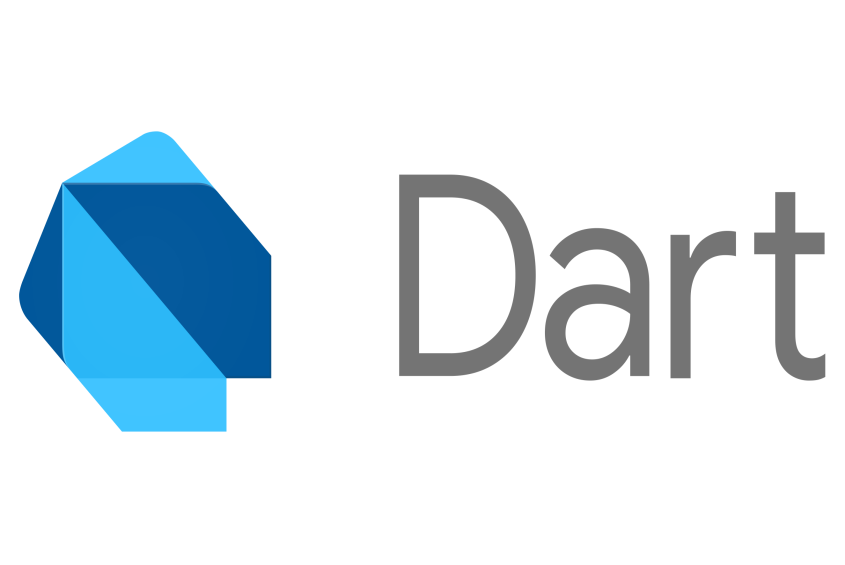
- Flutter sử dụng cơ chế Hot Reload cho phép lập trình viên xem ngay lập tức các thay đổi trong ứng dụng mà không cần khởi động lại. Nó hỗ trợ nhiều nền tảng như Android, iOS, Web, và Desktop. Flutter giúp tiết kiệm thời gian và công sức khi phát triển ứng dụng đa nền tảng, làm cho việc duy trì ứng dụng trở nên dễ dàng hơn.



# **Hình 1 : Flutter**

### Dart :

- Dart, ngôn ngữ lập trình của Flutter, được thiết kế để cung cấp hiệu suất cao và trải nghiệm phát triển tốt. Bên cạnh đó, cộng đồng của Flutter rất lớn và nhiệt tình, cung cấp nhiều gói và thư viện hữu ích để giúp lập trình viên xây dựng ứng dụng phức tạp và đa dạng.



# **Hình 2 : Dart**

- Với những ưu điểm vượt trội này, Flutter đã thu hút sự quan tâm và phát triển mạnh mẽ, trở thành một trong những công cụ phổ biến cho việc phát triển ứng dụng di động và đa nền tảng.

## **Mục đích đề tài**

- Xây dựng một ứng dụng To-Do sử dụng framework Flutter, tập trung vào việc thực hành lập trình đa nền tảng và hiểu rõ cách hoạt động của Flutter trong việc phát triển ứng dụng di động.

1. Thiết kế giao diện ứng dụng.
2. Xây dựng chức năng quản lý danh sách công việc.
3. Lưu trữ dữ liệu.
4. Kiểm thử và sửa lỗi.

## Đối tượng và phạm vi nghiên cứu

### Đối tượng nghiên cứu:

Đề tài "Flutter To-Do app" nhằm giúp những người mới làm quen với Flutter có cơ hội thực hành và hiểu rõ cách xây dựng ứng dụng cơ bản bằng framework này. Sử dụng ứng dụng để quản lý danh sách công việc.

### Phạm vi nghiên cứu:

* **Phạm vi chức năng ứng dụng :**

- Tạo, chỉnh sửa và xóa công việc trong danh sách.

- Hiển thị chi tiết công việc.

- Lưu trữ dữ liệu danh sách công việc.

- Đồng bộ dữ liệu giữa các thiết bị sử dụng ứng dụng.

* **Phạm vi kỹ thuật :**

- Sử dụng framework Flutter để xây dựng giao diện người dùng.

- Sử dụng ngôn ngữ lập trình Dart để triển khai logic ứng dụng.

- Sử dụng cơ chế State Management để quản lý trạng thái ứng dụng.

- Lưu trữ dữ liệu sử dụng cơ chế lưu trữ như SQLite hoặc Firebase.

# 

# Chương 2: CƠ SỞ LÝ LUẬN

## 2.1. Lý thuyết

### 2.1.1. **Tổng quan về Navigator** ?

Là công cụ quan trọng trong lĩnh vực phát triển ứng dụng di động. Thường được sử dụng trong các framework và thư viện lập trình đa nền tảng như React Native và Flutter để quản lý việc điều hướng giữa các màn hình (screens) trong ứng dụng di động.

* **Điều hướng (Navigation) :** Là quá trình chuyển đổi giữa các màn hình trong ứng dụng di động. Mỗi màn hình thường đại diện cho một phần nội dung cụ thể hoặc một chức năng trong ứng dụng.
* **Stack Navigator :** Navigator thường được triển khai dưới dạng một Stack Navigator, tức là sử dụng ngăn xếp (stack) để quản lý lịch sử điều hướng. Khi chuyển đổi từ một màn hình sang màn hình khác, màn hình mới được đẩy vào ngăn xếp, và khi người dùng thoát khỏi màn hình hiện tại, màn hình trước đó sẽ được pop ra khỏi ngăn xếp.
* **Thay đổi trạng thái (State Management) :** Một trong những thách thức lớn trong điều hướng là quản lý trạng thái của ứng dụng khi chuyển đổi giữa các màn hình. Điều này đặc biệt phức tạp khi các màn hình cần truy cập và chia sẻ dữ liệu chung.
* **Thao tác điều hướng** : Có một số phương thức cơ bản để thực hiện điều hướng trong Navigator.

- Push : Đẩy một màn hình mới vào ngăn xếp.

- Pop : Lấy ra màn hình khỏi ngăn xếp (quay lại màn hình trước đó).

- Replace : Thay thế màn hình hiện tại bằng màn hình mới.

- PopToTop : Quay về màn hình đầu tiên trong ngăn xếp.

**Navigator** : Thường được tối ưu để hoạt động trên nhiều nền tảng di động khác nhau như Android và iOS, giúp phát triển ứng dụng đa nền tảng dễ dàng hơn.

### 2.1.2. **Tổng quan về Router** ?

- Cho phép quản lý các tuyến (routes) của ứng dụng, đồng thời điều hướng và điều phối hiển thị giữa các màn hình khi người dùng tương tác với ứng dụng.

- Router giúp điều hướng giữa các màn hình dễ dàng và đảm bảo tính linh hoạt và hiệu quả trong phát triển ứng dụng đa nền tảng.



**Hình 3 : Router**

* **Điều hướng (Routing) :** Router quản lý việc chuyển đổi giữa các màn hình trong ứng dụng. Mỗi màn hình thường đại diện cho một tuyến hoặc một phần nội dung cụ thể trong ứng dụng.
* **Tuyến (Route):** Một tuyến là một định danh cho một màn hình hoặc một phần nào đó của ứng dụng. Các tuyến thường được định nghĩa trong Router và ánh xạ với các thành phần UI tương ứng.
* **Thay đổi trạng thái (State Management):** Router thường liên quan chặt chẽ đến việc quản lý trạng thái của ứng dụng khi người dùng di chuyển giữa các màn hình. Dữ liệu và trạng thái của ứng dụng có thể được truyền qua các tuyến để đảm bảo nhất quán và chia sẻ thông tin giữa các màn hình.
* **Phương thức điều hướng:** Router cung cấp các phương thức để thực hiện việc điều hướng giữa các màn hình. Các phương thức này thường bao gồm:

- Navigate: Điều hướng đến màn hình mới.

- Go back: Quay lại màn hình trước đó.

- Replace: Thay thế màn hình hiện tại bằng màn hình mới.

- Router thường được thiết kế để hoạt động trên nhiều nền tảng di động khác nhau như Android và iOS, giúp phát triển ứng dụng đa nền tảng một cách dễ dàng và hiệu quả.

- Qua đó Router giúp tăng tính mô đun và quản lý ứng dụng di động một cách hiệu quả, giúp nhà phát triển dễ dàng mở rộng và duy trì ứng dụng trong quá trình phát triển và bảo trì.

### 2.1.3. **Di chuyển dữ liệu giữa các màn hình ?**

* **Truyền thông qua các tham số khi điều hướng :**

- Khi điều hướng từ một màn hình A đến màn hình B, có thể truyền các tham số kèm theo yêu cầu điều hướng. Các tham số này có thể là các giá trị đơn giản như : số nguyên, chuỗi, boolean, hoặc là các đối tượng phức tạp.

- **Ví dụ :** Truyền một ID sản phẩm từ màn hình danh sách sản phẩm (màn hình A) sang màn hình chi tiết sản phẩm (màn hình B) để hiển thị thông tin chi tiết về sản phẩm đó.

* **Sử dụng trạng thái toàn cục (Global State) :**

- Một số framework và thư viện cung cấp cơ chế trạng thái toàn cục, cho phép lưu trữ dữ liệu mà các màn hình có thể truy cập và chia sẻ thông qua các thành phần khác nhau trong ứng dụng.

- **Ví dụ :** Sử dụng Redux trong React Native hoặc Provider/Consumer trong Flutter để quản lý trạng thái toàn cục và chia sẻ dữ liệu giữa các màn hình.

* **Lưu trữ dữ liệu tạm thời :**

- Có thể lưu trữ dữ liệu tạm thời trong bộ nhớ đệm (cache) của ứng dụng hoặc lưu trữ cục bộ (local storage). Khi chuyển đổi giữa các màn hình, có thể truy xuất dữ liệu từ bộ nhớ đệm hoặc lưu trữ cục bộ mà không cần truyền dữ liệu trực tiếp giữa các màn hình.

* **Sử dụng Event Bus hoặc Pub/Sub :**

- Một cách khác để chia sẻ dữ liệu giữa các màn hình là thông qua Event Bus hoặc Pub/Sub mechanism. Đây là cơ chế cho phép các thành phần trong ứng dụng gửi và lắng nghe các sự kiện (events) và dữ liệu có thể được chuyển đổi giữa các màn hình thông qua các sự kiện này.

* **Truyền dữ liệu qua hệ thống quản lý URL (URL Parameters) :**

- Có thể được truyền dưới dạng tham số trong URL. Màn hình mới có thể trích xuất và sử dụng dữ liệu từ URL để hiển thị thông tin phù hợp.

- Cách di chuyển dữ liệu giữa các màn hình có thể khác nhau tùy thuộc vào framework và thư viện đang sử dụng. Chọn cách phù hợp và tối ưu cho ứng dụng là điều quan trọng để đảm bảo tính linh hoạt và hiệu quả trong phát triển ứng dụng đa nền tảng.

### 2.1.4. **Lập trình bất đồng bộ với Future, async, await**

- Là một khái niệm quan trọng trong ngôn ngữ lập trình để xử lý các tác vụ mà thời gian thực thi không thể dự đoán được. như gửi yêu cầu mạng, truy vấn cơ sở dữ liệu hoặc xử lý dữ liệu lớn, các tác vụ đòi hỏi thời gian chờ lâu dài. Điều này giúp duy trì hiệu suất ứng dụng, tránh tình trạng đứng chương trình trong quá trình chờ đợi.

- Trong Flutter (và ngôn ngữ Dart), để thực hiện lập trình bất đồng bộ sử dụng cơ chế Future, async và await.



**Hình 4 : Future, async, await**

* **Future :** Là một đối tượng đại diện cho một giá trị hoặc một lỗi sẽ được trả về sau khi một tác vụ bất đồng bộ hoàn thành. Future có thể được xem như một hợp đồng rằng bạn sẽ nhận được kết quả sau khi tác vụ hoàn thành.
* **Async :** Từ khóa "async" được sử dụng để khai báo một hàm là bất đồng bộ. Hàm được khai báo bằng async có khả năng chứa các tác vụ bất đồng bộ và sử dụng từ khóa "await" để đợi kết quả của các Future.
* **Await :** Từ khóa "await" được sử dụng trong hàm bất đồng bộ để đợi kết quả của một Future. Khi gặp từ khóa "await", thực thi của hàm tạm dừng và chờ đợi cho đến khi Future hoàn thành và trả về kết quả.
* **Sử dụng trong Lập trình Đa nền tảng cho ứng dụng di động :**

Trong việc phát triển ứng dụng di động đa nền tảng, việc sử dụng lập trình bất đồng bộ với Future, async và await có vai trò quan trọng trong các tình huống như :

**- Gửi yêu cầu mạng :** Khi gửi yêu cầu đến máy chủ để lấy dữ liệu, việc sử dụng Future, async và await giúp tránh chặn luồng chính và cho phép ứng dụng vẫn tiếp tục hoạt động trong thời gian chờ đợi dữ liệu trả về.

**- Xử lý tác vụ nặng :** Trong trường hợp xử lý dữ liệu lớn, tính toán phức tạp, lập trình bất đồng bộ giúp tránh làm đứng chương trình và tạo ra trải nghiệm tốt hơn cho người dùng.

**-** **Truy vấn cơ sở dữ liệu :** Khi thực hiện truy vấn đến cơ sở dữ liệu, việc sử dụng bất đồng bộ giúp đảm bảo ứng dụng không bị đứng do thời gian truy vấn kéo dài.

**-** **Ví dụ :** Trong một ứng dụng To-Do, khi người dùng thêm công việc mới, việc lưu trữ dữ liệu có thể là một tác vụ bất đồng bộ để đảm bảo ứng dụng vẫn phản hồi nhanh chóng mà không bị đóng băng trong quá trình lưu trữ dữ liệu.

### 2.1.5. **Firebase**

- Firebase là một nền tảng phát triển ứng dụng di động và web do Google phát triển, cung cấp một loạt các dịch vụ và công cụ để giúp các nhà phát triển xây dựng và quản lý ứng dụng một cách dễ dàng và hiệu quả. Firebase giúp giảm bớt công việc phát triển từ đầu và tập trung vào việc xây dựng các tính năng chất lượng cho ứng dụng.



**Hình 5 : Firebase**

* **Các dịch vụ và khả năng chính của Firebase bao gồm :**

1. **Realtime Database :** Dịch vụ cơ sở dữ liệu thời gian thực, cho phép lưu trữ và đồng bộ dữ liệu giữa các thiết bị một cách tức thì. Thích hợp cho các ứng dụng yêu cầu đồng bộ dữ liệu nhanh chóng như ứng dụng chat.
2. **Cloud Firestore :** Một cơ sở dữ liệu NoSQL linh hoạt hơn, cho phép lưu trữ dữ liệu theo tài liệu và cung cấp tính năng truy vấn mạnh mẽ. Firestore thích hợp cho việc xây dựng các ứng dụng có cấu trúc dữ liệu phức tạp.
3. **Authentication :** Dịch vụ xác thực người dùng, hỗ trợ đăng nhập bằng nhiều phương thức như email, Google, Facebook và nhiều phương thức xác thực khác. Điều này giúp tiết kiệm thời gian và tạo trải nghiệm đăng nhập thuận tiện cho người dùng.
4. **Cloud Storage :** Dịch vụ lưu trữ dữ liệu đa phương tiện như hình ảnh, video, tệp tin. Firebase Storage cung cấp việc quản lý và truy cập dữ liệu nhanh chóng và bảo mật.
5. **Cloud Functions :** Cho phép viết mã logic và triển khai nó lên máy chủ của Firebase, giúp xử lý các tác vụ như gửi thông báo đẩy, xác thực dữ liệu, hoặc tích hợp các dịch vụ bên ngoài.
6. **Hosting :** Dịch vụ lưu trữ và triển khai ứng dụng web tĩnh. Firebase Hosting giúp bạn dễ dàng đưa ứng dụng web lên mạng và quản lý nó.
7. **Analytics :** Cung cấp thông tin phân tích về cách người dùng tương tác với ứng dụng. Dữ liệu này giúp bạn hiểu rõ hơn về người dùng và tối ưu hóa trải nghiệm của họ.
8. **Cloud Messaging :** Cho phép gửi thông báo đẩy tới các thiết bị di động và web. Firebase Cloud Messaging giúp tương tác với người dùng và cung cấp thông báo về các sự kiện quan trọng.
9. **Authentication :** Cung cấp các công cụ quản lý và bảo vệ dữ liệu người dùng, cho phép xác thực người dùng qua nhiều phương thức khác nhau.

- Firebase cung cấp tích hợp tốt với nhiều ngôn ngữ lập trình và nền tảng khác nhau, bao gồm cả Android, iOS, web, và cả các nền tảng khác thông qua API. Điều này giúp tạo ra ứng dụng đa nền tảng dễ dàng và hiệu quả.

## 2.2. Kĩ thuật

### 2.2.1. Ngôn ngữ sử dụng :

- Ngôn ngữ lập trình Dart.

### **2.2.2. Framework :**

- Flutter.

### 2.2.3. Công cụ:

- Android Studio.

### 2.2.4. Thư viện giao diện:

- Widgets Tùy chỉnh.

- Cupertino Widgets.

- Material Design.

### 2.2.5. Hệ quản trị cơ sở dữ liệu:

# - Firebase.

# Chương 3: CÀI ĐẶT THỬ NGHIỆM

## **3.1. Nghiên cứu đặc tả chức năng, thiết kế cơ sở dữ liệu**

### 3.1.1. Đặc tả chức năng :

- Sơ đồ Use Case chi tiết này mô tả các hoạt động quan trọng trong ứng dụng To-Do và cách người dùng tương tác với ứng dụng.

1. Người dùng đăng nhập :

Mô tả : Người dùng sử dụng tên đăng nhập email và mật khẩu để đăng nhập vào ứng dụng.

- Người dùng mở ứng dụng.

- Ứng dụng hiển thị giao diện đăng nhập.

- Người dùng nhập tên đăng nhập email và mật khẩu.

- Người dùng nhấn nút "Đăng nhập".

Hệ thống kiểm tra thông tin đăng nhập và cho phép truy cập nếu thông tin chính xác.

1. Người dùng xem danh sách công việc :

Mô tả : Người dùng xem danh sách các công việc mà họ đã tạo.

- Người dùng đã đăng nhập vào ứng dụng.

- Người dùng chọn mục "Danh sách công việc".

- Ứng dụng hiển thị danh sách công việc đã tạo bởi người dùng.

1. Người dùng thêm công việc mới :

Mô tả : Người dùng thêm một công việc mới vào danh sách.

- Người dùng đã đăng nhập vào ứng dụng.

- Người dùng chọn mục "Thêm công việc".

- Ứng dụng hiển thị giao diện để nhập thông tin công việc (tiêu đề, mô tả, loại nhiệm vụ,…).

- Hệ thống lưu thông tin công việc vào cơ sở dữ liệu và cập nhật danh sách công việc.

1. Người dùng chỉnh sửa công việc :

Mô tả : Người dùng sửa lại một công việc là đã hoàn thành.

- Người dùng đã đăng nhập vào ứng dụng và đã xem danh sách công việc.

- Người dùng chọn một công việc từ danh sách.

- Ứng dụng hiển thị thông tin chi tiết của công việc.

- Bắt đầu chỉnh sửa các mục thông tin công việc (tiêu đề, mô tả, loại nhiệm vụ,…)

- Hệ thống cập nhật trạng thái công việc và hiển thị danh sách công việc cập nhật.

1. Người dùng xóa công việc :

Mô tả : Người dùng xóa một công việc khỏi danh sách.

- Người dùng đã đăng nhập vào ứng dụng và đã xem danh sách công việc.

- Người dùng chọn một công việc từ danh sách.

- Ứng dụng hiển thị thông tin chi tiết của công việc.

- Người dùng chọn tùy chọn xóa công việc và xác nhận.

- Hệ thống xóa công việc khỏi cơ sở dữ liệu và cập nhật danh sách công việc còn lại.

1. Người dùng đăng xuất :

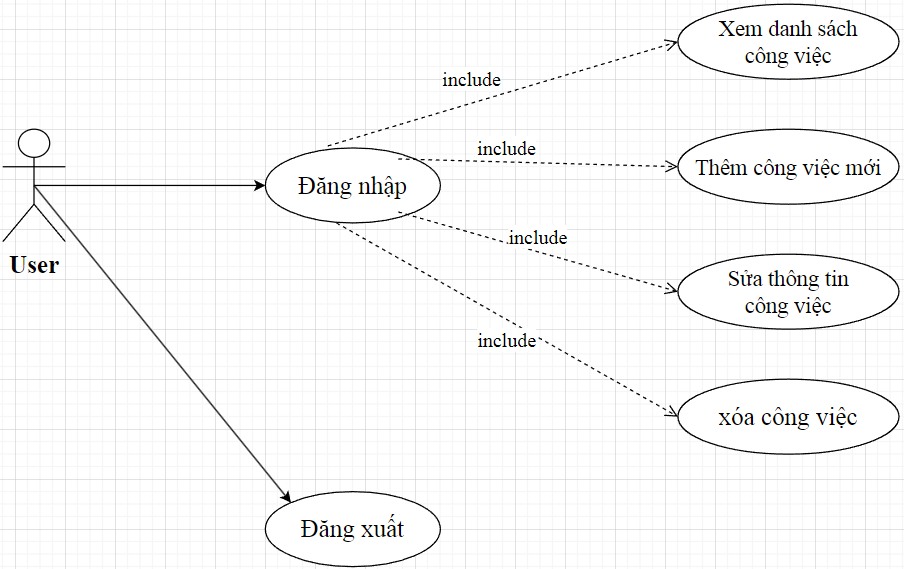
Mô tả : Người dùng đăng xuất khỏi tài khoản của mình.

- Người dùng đã đăng nhập vào ứng dụng.

- Người dùng chọn tùy chọn đăng xuất.

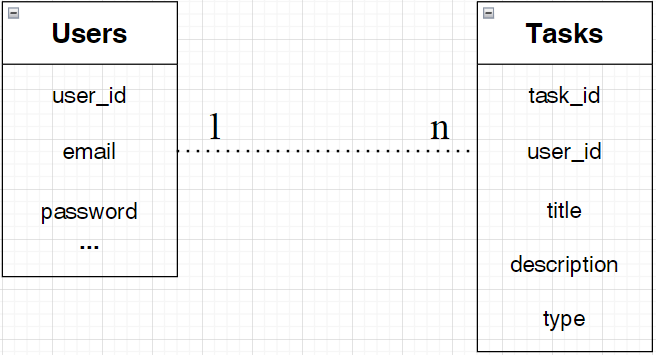
- Ứng dụng xóa phiên đăng nhập hiện tại và quay lại màn hình đăng nhập.

### 3.1.2. Lược đồ use case:



**Hình 6 : Lược đồ Use - case**

### 3.1.3. ERD:



**Hình 7 : ERD**

**Thực thể Người dùng (Users) :** Chứa thông tin về người dùng như

- user\_id : Định danh duy nhất cho người dùng..

- Email : Địa chỉ email của người dùng.

- Password : Mật khẩu đăng nhập của người dùng

…

**Thực thể Công việc (Tasks) :** chứa thông tin về công việc như

- task\_id : Định danh duy nhất cho công việc.

- user\_id : Liên kết đến user\_id trong thực thể Người dùng, xác định người dùng tạo công việc.

- Title : Tiêu đề của công việc.

- Description : Mô tả chi tiết công việc.

- Type : Loại công việc.

**Mối quan hệ giữa các thực thể : ( 1 - n )**

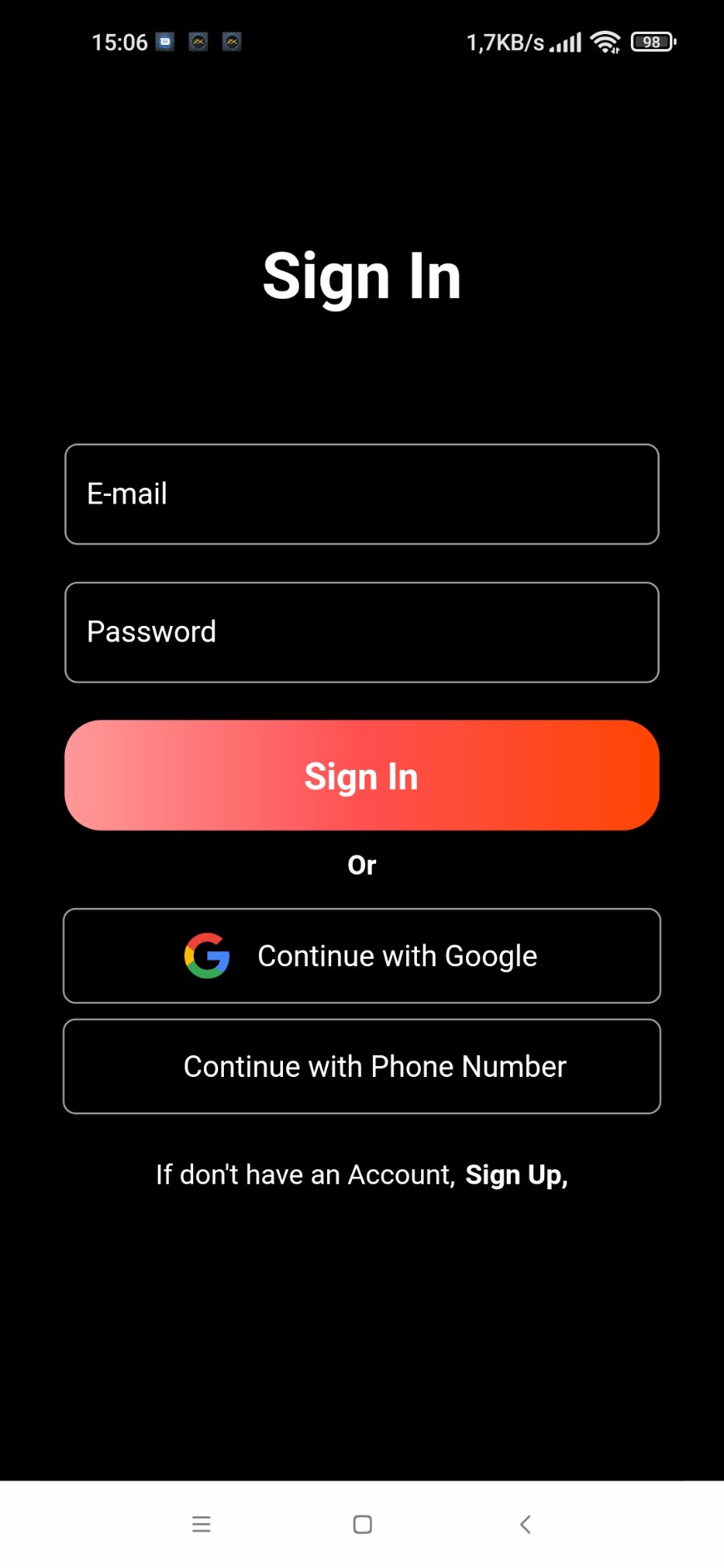
- Một người dùng (Users) có thể có nhiều công việc (Tasks).

- Mỗi công việc (Tasks) thuộc về một người dùng (Users).

- Sơ đồ ERD này giúp bạn hiểu cấu trúc dữ liệu cơ sở dữ liệu của ứng dụng To-Do và quan hệ giữa các thực thể. Điều này có thể giúp bạn xây dựng cơ sở dữ liệu phù hợp để lưu trữ và quản lý danh sách công việc trong ứng dụng của mình.

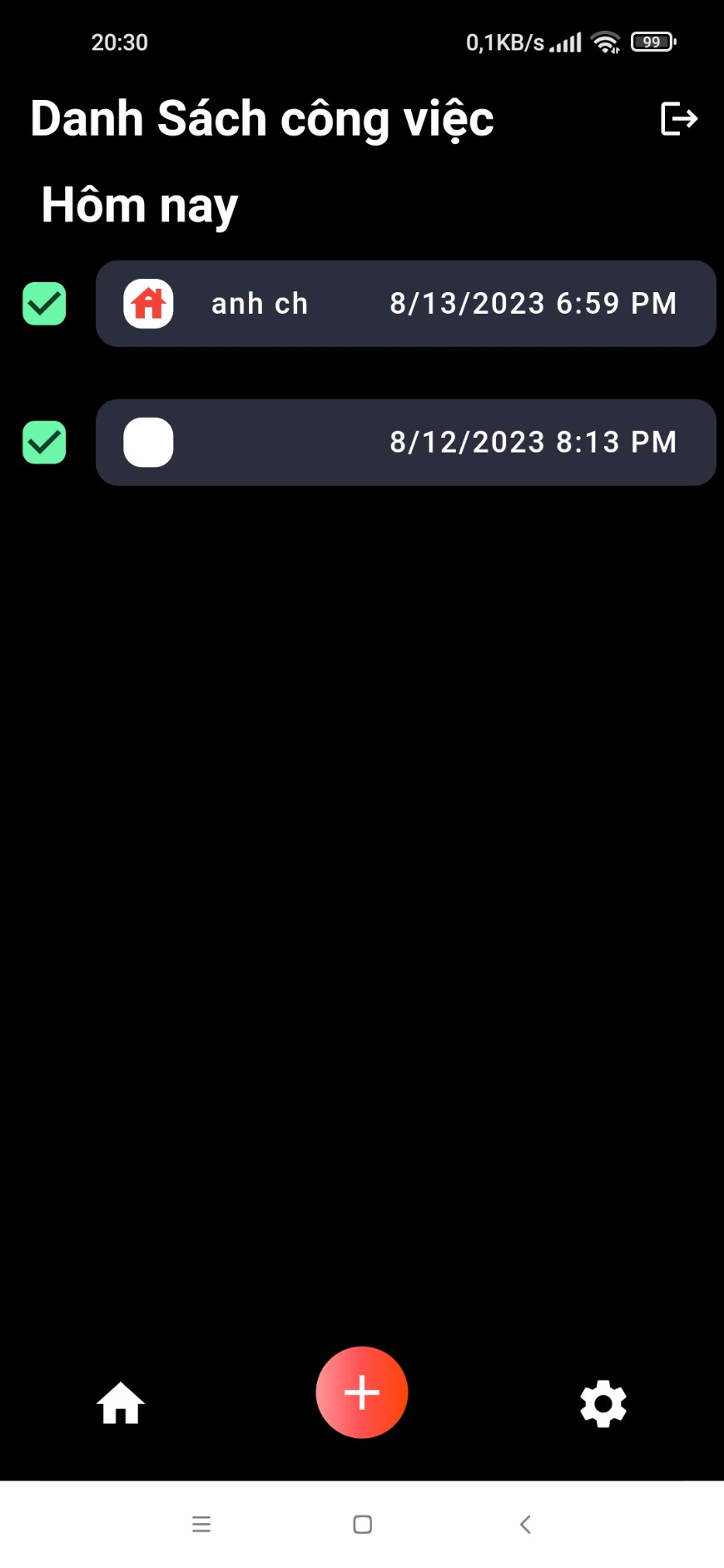
## 3.2. **Chi tiết và demo hình ảnh các chức năng** :

### 3.2.1. Giao diện đăng nhập :

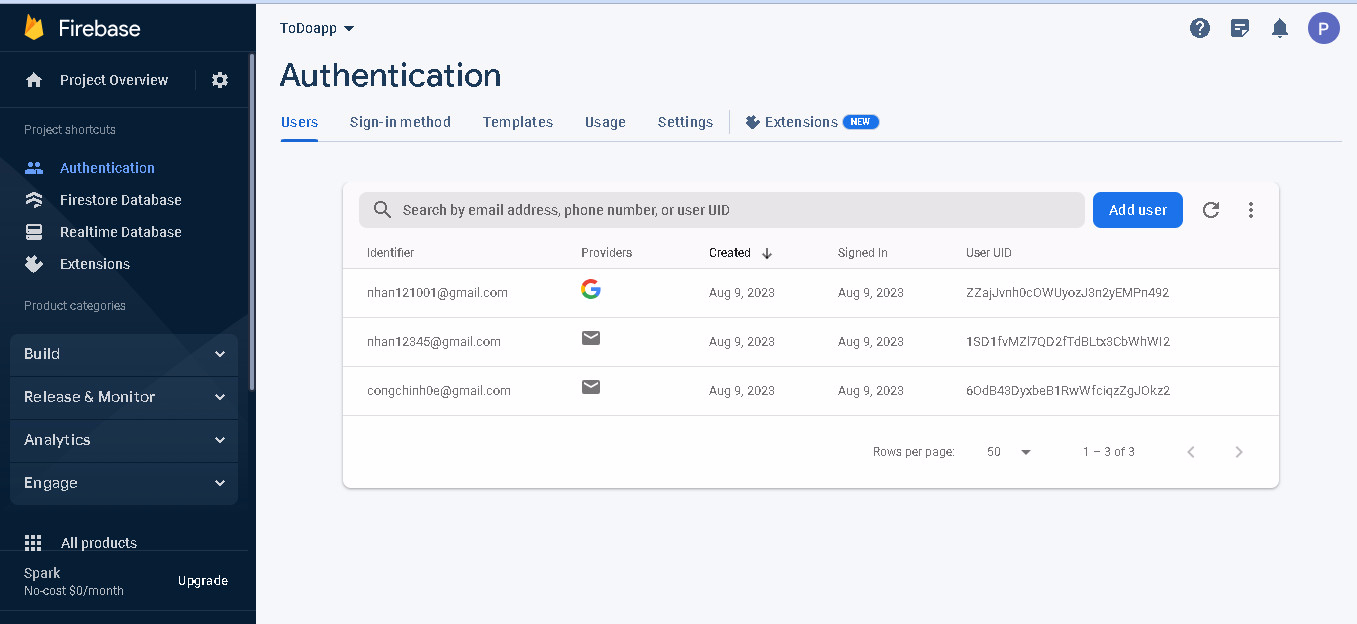


**Hình 8 : Giao diện trang chủ**

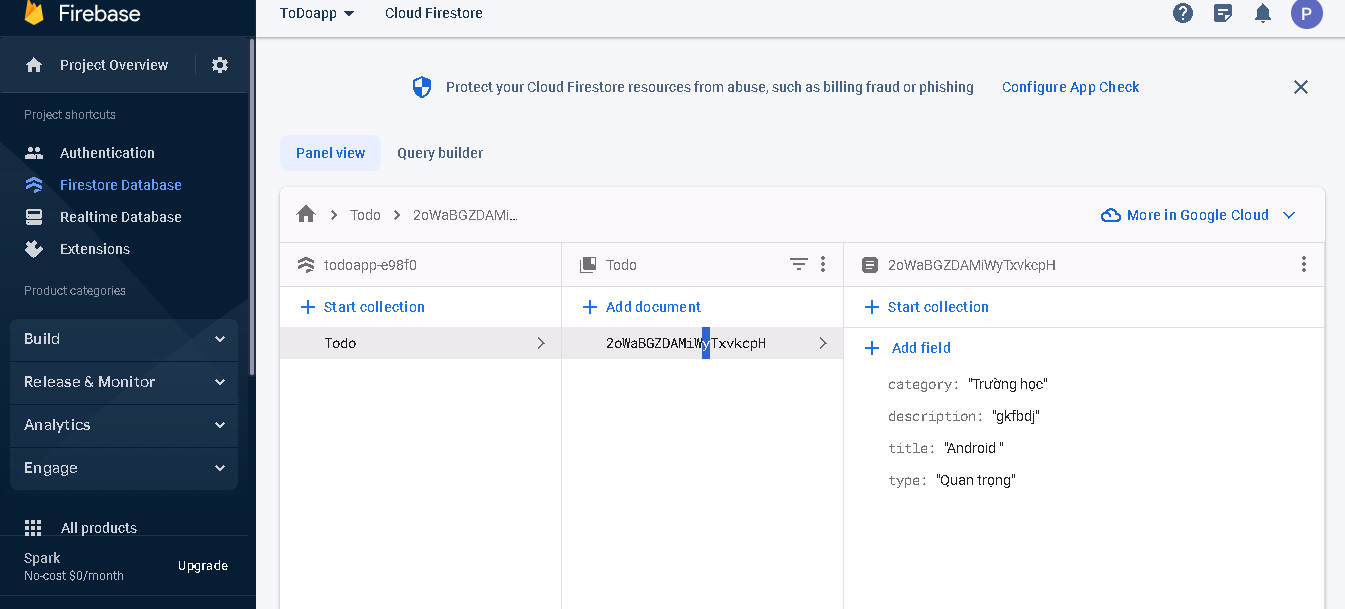
### 3.2.2. Giao diện trang :



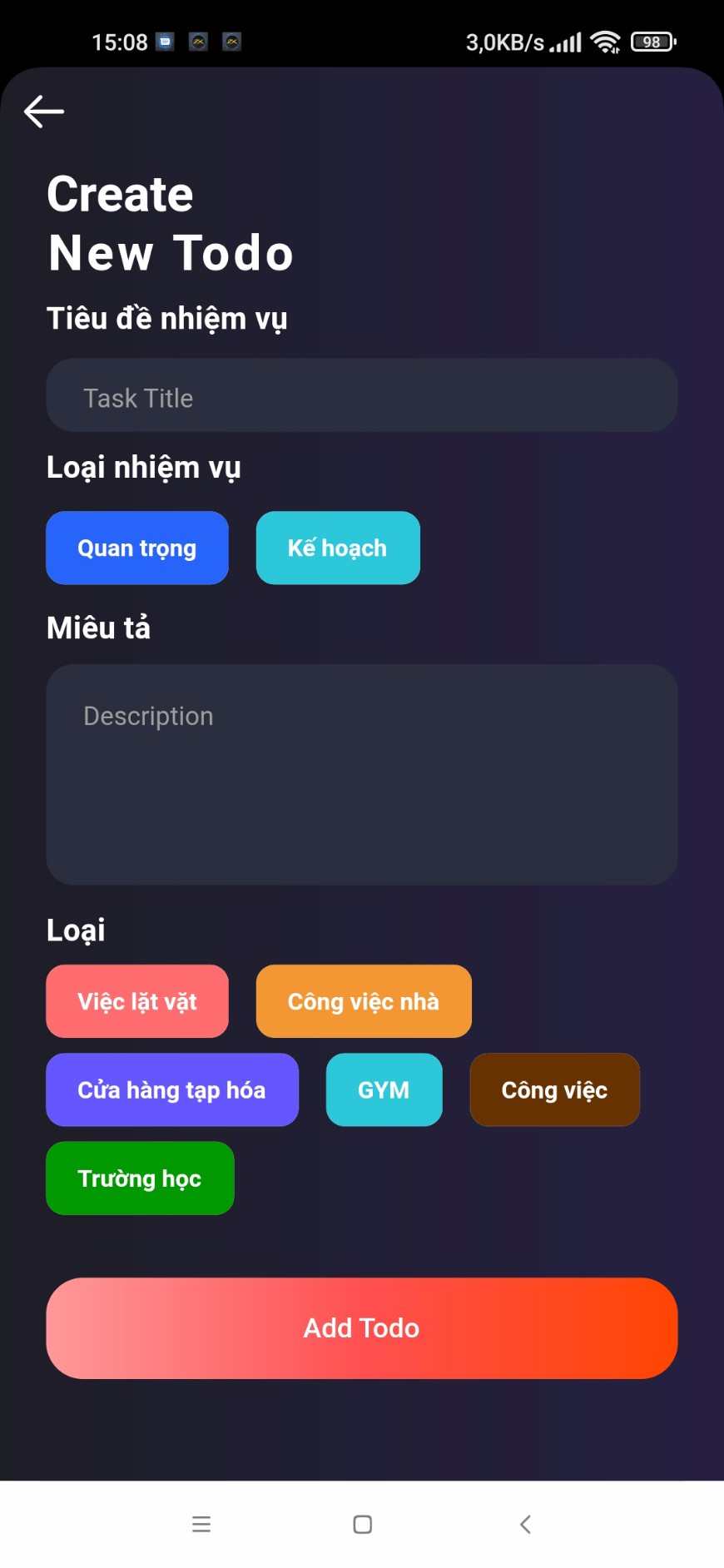
**Hình 9 : Giao diện trang danh sách công việc**



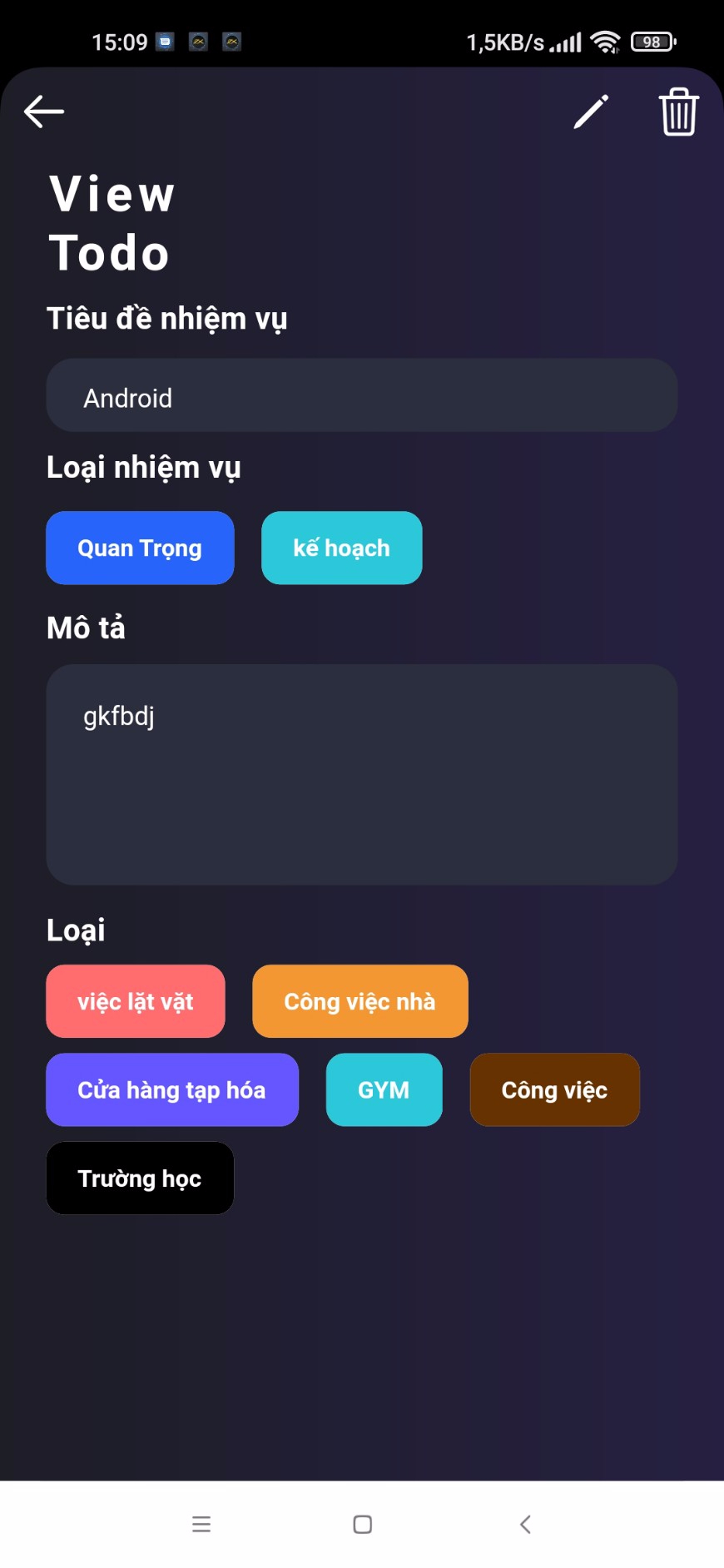
**Hình 10 : Giao diện trang Authentication**



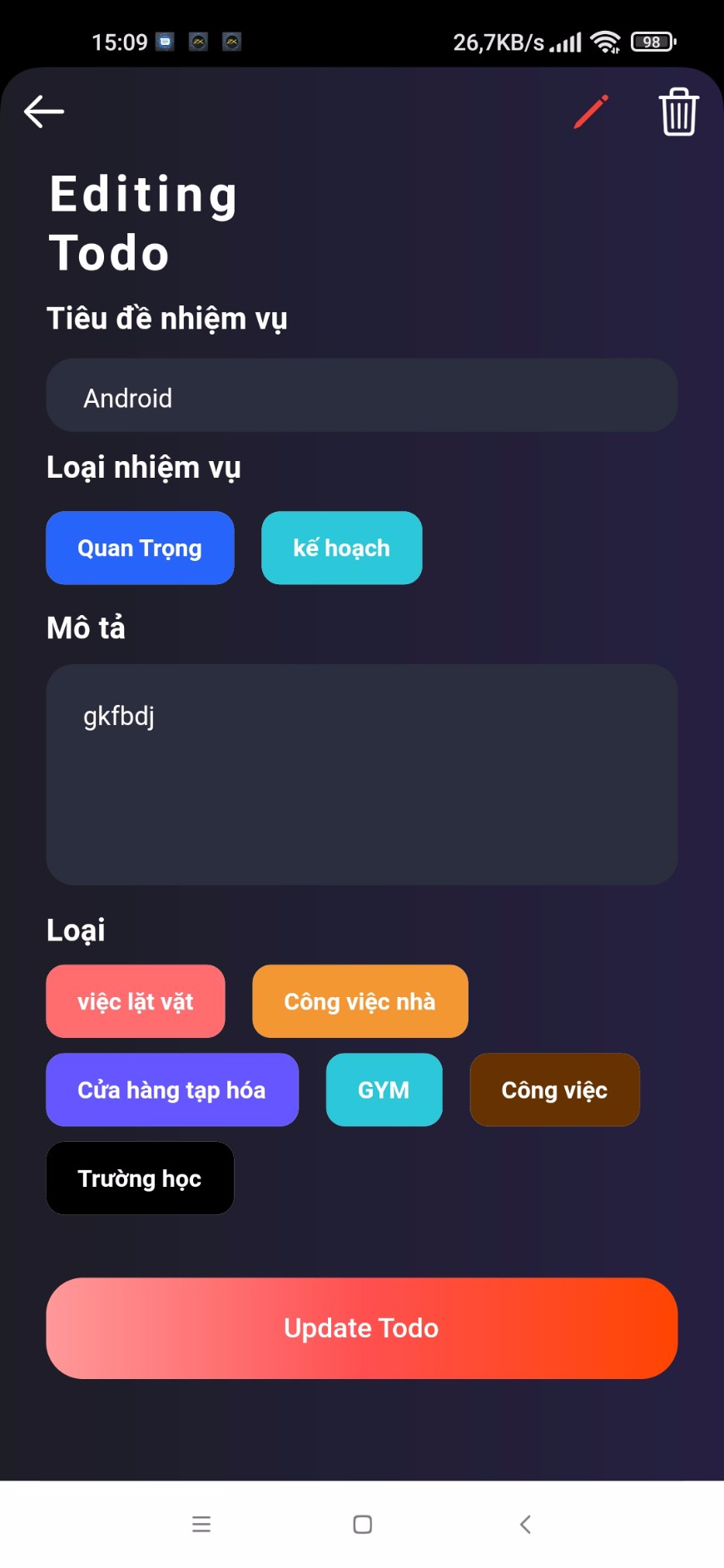
**Hình 11 : Giao diện trang Firestore Database**



**Hình 12 : Giao diện trang thêm công việc**



**Hình 13 : Giao diện trang chi tiết công việc**



**Hình 14 : Giao diện sửa thông tin công việc**

# Chương 4: KẾT LUẬN

## 4.1. Kết quả đạt được:

- Trong báo cáo này, đã thực hiện nghiên cứu và phát triển ứng dụng "Flutter To-Do app" trong môn học Công Nghệ Lập Trình Đa Nền Tảng Cho Ứng Dụng Di Động. Đây là một ứng dụng nhỏ nhưng mang tính ứng dụng thực tế, giúp người dùng quản lý danh sách công việc một cách hiệu quả.

- Trong giai đoạn nghiên cứu, nhóm em đã tìm hiểu về công nghệ Flutter - một nền tảng phát triển ứng dụng đa nền tảng của Google. Flutter cho phép xây dựng giao diện người dùng đẹp và tương tác trên nhiều nền tảng khác nhau một cách dễ dàng. Nhóm em cũng đã tìm hiểu về các khái niệm cơ bản như lập trình bất đồng bộ.

- Đã thiết kế và phát triển ứng dụng "Flutter To-Do app". Ứng dụng cho phép người dùng đăng nhập, xem danh sách công việc, thêm mới, đánh dấu hoàn thành và xóa các công việc. Nhóm em đã sử dụng thư viện giao diện Material Design để tạo giao diện đẹp và thân thiện với người dùng. Đồng thời, chúng ta cũng đã tích hợp Firebase để lưu trữ dữ liệu công việc và đảm bảo tính đồng bộ trên các thiết bị.

- Tóm lại, đề tài "Flutter To-Do app" đã giúp nhóm em áp dụng kiến thức về lập trình đa nền tảng và phát triển ứng dụng thực tế trên nền tảng Flutter. Qua quá trình này, nhóm đã nắm vững các khái niệm quan trọng và có kinh nghiệm trong việc xây dựng ứng dụng di động sử dụng công nghệ hiện đại.

- Trong thời gian nghiên cứu, xây dựng chương trình, nhóm em đã hết sức cố gắng làm việc với sự giúp đỡ tận tình của giảng viên. Chương trình đã đạt được kết quả nhất định. Tuy nhiên với thời gian ngắn, trình độ và kinh nghiệm còn hạn chế nên chương trình vẫn còn nhiều thiếu sót.

- Một lần nữa nhóm xin chân thành cảm ơn thầy **Nguyễn Thanh Truyền** đã tận tình giúp đỡ nhóm em trong suốt thời gian thực hiện đồ án tốt nghiệp. Nhóm xin chân thành cảm ơn thầy.

# DANH MỤC TÀI LIỆU THAM KHẢO

**Tiếng Anh:**

1. **English books Max30** *Flutter for Beginners* **<https://ebooksmax30.com/products/flutter-for-beginners-an-introductory-guide-to-building-cross-platform-mobile-applications-with-flutter-2-5-and-dart-2nd-edition?variant=1106104199&source=googleshop>**

# [2] Packt Publishing*Flutter Projects*

**https://fado.vn/us/amazon/flutter-projects-a-practical-project-based-guide-to-building-real-world-cross-platform-mobile-applications-and-games-1838647775.html?\_mzfrom=gg7721&utm\_source=google&utm\_medium=cpc&ref\_id=googlepermax&gclid=EAIaIQobChMIx\_T63OvOgAMVDQNyCh1fPA8lEAQYBSABEgLyqfD\_BwE**

**[3] Diagram** *Công cụ hỗ trợ thiết kế giao diện* https://diagrams.net/